(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 July 2001 (12.07.2001)

PCT

(10) International Publication Number
WO 01/50320 A1

(51) International Patent Classification⁷: G06F 17/00, 17/30

(21) International Application Number: PCT/US00/35169

(22) International Filing Date: 21 December 2000 (21.12.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/173,779 30 December 1999 (30.12.1999) US
09/592,012 12 June 2000 (12.06.2000) US

(71) Applicant: AUCTIONWATCH.COM, INC. [US/US]; Suite 100, 851 Traeger Road, San Bruno, CA 94066 (US).

(72) Inventors: COUSINS, Robert, E.; 14330 Cordwood Court, Saratoga, CA 95070 (US). SLAYTON, Marc, A.; 219 Missouri Street, San Francisco, CA 94107 (US). MARGOLIN, Benjamin; Apartment 2E, 1500 Sherman Avenue, Burlingame, CA 94010 (US).

(74) Agents: RAO, Dana, S. et al.; Fenwick & West LLP, Two Palo Alto Square, Palo Alto, CA 94306 (US).

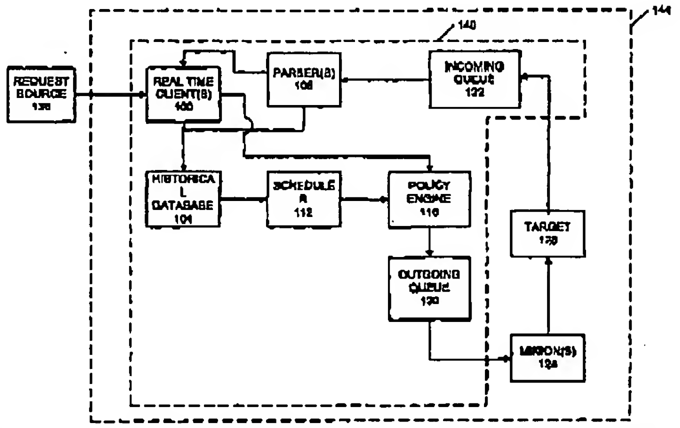
(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: MINIMAL IMPACT CRAWLER



(57) Abstract: A system and method are provided that provide a minimal impact crawler (144) for searching and retrieving information on a distributed network. A policy engine (116) is provided that receives a request for a specific item and assembles policies for the target site containing information about the specific item. The policies are rules that determine the crawl (144) of a target site (128). The crawler (144) applies the policies to schedule crawls (144) of the target site (128) and stores data retrieved from the crawl (144) into a historical database (104) allowing future requests to be satisfied from the data stored in the database. A scheduling engine is implemented to automatically schedule crawls (144) at the beginning of an auction and at the end of an auction to minimize the number of crawls (144) on an auction site. The crawler (144) employs a plurality of minions (144) to retrieve crawl (144) requests and crawl (144) the target web sites (128) to obtain the necessary data.

WO 01/50320 A1

MINIMAL IMPACT CRAWLER

RELATED APPLICATION

The present application claims the benefit of the filing date of U.S. Provisional application serial number 60/173,779.

FIELD OF THE INVENTION

The present invention relates to database search technology, and more specifically to a crawler for accessing data on a remote web site.

BACKGROUND OF THE INVENTION

The Web hosts millions of different web sites. Many of those web sites manage collections of data in support of services they sell or offer for free to users. However, given the enormous number of web sites, it is difficult for individual web sites to become known to users. Therefore, third-party crawlers have been developed (such as Alta Vista ™, WebCrawler ™) that provide data gathering, searching, and retrieval services to allow users or Internet-related applications to locate desired data from the millions of web sites.

Crawlers gather this information by using the Internet in a process known as crawling. Essentially, computers of the crawler take on the role of browsing users and access the target web sites using the standard web protocols (TCP/IP, HTTP, etc.) over the internet. As crawlers are computer-driven, they can access large amounts of data very quickly, and they can perform crawls continuously. Web sites typically view crawlers as a beneficial process because without crawlers, many web sites would never be encountered by the average user.

One important category of web sites that manage large amounts of data is the field of online auctions, such as those provided by eBay.com, Yahoo.com, and Amazon.com. Web sites that provide online auctions must maintain large databases of information relating to the various auctions being conducted through their web site, such as bid price, start/stop times, item information, seller information, buyer information, and the like. This information must be organized such that potential bidders can find and search the auction site easily. If auction information is not easily accessible, potential bidders will not be able to view an item for sale, and thus the final selling price of the item will not reflect its actual market value. If sellers feel they are unable to achieve the market price for their goods at a particular auction site, they will look for a different site at which to market their goods. Thus, it is important for the economic success of an auction site that as many bidders can view an item as possible, and as easily as possible.

Therefore, a crawling service for auction sites provides a very useful benefit to both the buyers, who are able to view auctions from many sites that they would ordinarily encounter, and to the sellers, who are exposed to more buyers and therefore will be able to generate the highest price for their goods. Auction sites typically assign auctions a unique number by the auction host such as *211266203*. This translates to a standardized URL for referring to the auction. A simple crawler could merely start with auction number one, fetch the information for that auction and then move to auction number two , and stopping when the final auction has been reached. Repeating this process at regular intervals will guarantee that the crawling company will have all information as current as the last crawl period for a particular auction site.

Unfortunately, there are a number of problems associated with this method. First, this method places a huge burden on the crawled site. By persistently fetching the same pages without any insight as to what pages will change and when they will change, the crawler must traverse many pages which will ultimately yield no additional (or new) information. This will eventually increase the load on the crawled site, consume bandwidth, and decrease the crawled site's ability to serve requests. The crawled site may be forced to add expensive equipment and take other steps to meet this demand. Second, this method places a huge burden on the crawling site. The processing power required to fetch and parse these pages is substantial and the network bandwidth required could be enormous. Moreover, the conventional method is single threaded. In the example above, only one page could be fetched at a time. This implies that if it takes 1 minute to fetch a given auction, no more than 1440 auctions could be crawled in a given 24 hour period. Clearly, there are more auctions than this, so a single threaded approach is unlikely to cover the required auctions rendering the crawl useless.

Therefore, what is needed is a massively parallel, distributed, intelligent crawling system which minimizes the burden on both the crawler and a target site to be crawled. Although the discussion herein uses the concept of auction sites as an exemplary embodiment of the present invention, the present invention should not be viewed as being limited to auctions. In reality, the same technology can be used to drive any crawler-related use. For example, the present invention applies to commodity trading sites, classified advertisement sites, book finding sites, or other sites that maintain and search large amounts of information.

SUMMARY OF INVENTION

In accordance with the present invention, a system, method, and system are provided that provide a minimal impact crawler for searching for and retrieving information on a distributed network. In a preferred embodiment, a policy engine is provided that receives a request for a specific item and assembles polices for the target site containing information about the specific item. Policies are maintained in databases sorted by different categories such as target site, time of day, location of target site, etc. The policies themselves include rules identified for target sites such as "Do not crawl between 9:00 a.m. and 5:00 p.m." or "Do not crawl if web site is shut down", or "Never crawl." The policies allow the crawler to tailor its crawl to the requirements and preferences of the crawled site. This allows the crawler and the crawled site to create an optimal crawling schedule to satisfy the target site's need to be accessible to as many people as possible and the target site's need to avoid overly burdensome traffic.

In another embodiment, a crawler maintains a historical database to further minimize the impact of a crawl on a crawled site. In this embodiment, after a crawl, either scheduled or by request, the crawler stores data retrieved from the crawl into a historical database. Then, when a scheduled or user-demanded crawl is to occur, the crawler first checks the historical database to determine whether the request can be satisfied from the data stored in the database. In a preferred embodiment, the crawler maintains date and time information for data gathered from a crawl, and one of the policies only permits data stored in the database to be used to satisfy a request if the data was collected within a predefined period of time. For example, if a user-demanded crawl is to be initiated of a target site, but a scheduled crawl had been initiated within the last hour, the crawler will terminate the user-demanded crawl because the data in the historical database is sufficiently recent to satisfy the user request. Thus, by maintaining a historical database, the crawler of the present invention can avoid crawls completely by satisfying a request from the historical database, thereby minimizing traffic on a target site.

In a further embodiment for an auction-crawling implementation of the present invention, a scheduling engine is implemented to automatically schedule crawls at the beginning of an auction and at the end of an auction. As much of the information provided in an auction is unchanging, these two crawls provide the majority of the information needed to represent an auction, while minimizing the number of crawls on the auction site. For example, item information and seller information do not change during the life of an auction. Therefore, the initial crawl can provide this information, and then no further crawl is necessary unless a user request for price updates is received. Also, the closing price and winning buyer information does not change after the auction has ended, and therefore the final crawl can retrieve this information. Thus, by tailoring a crawler to an auction's unique characteristics, the crawler of the present invention can minimize the impact of crawling on an auction site.

The crawler of the present invention also integrates improved crawling technology into its minimal impact crawl. First, the crawler uses an output queue for crawl requests in which the crawl requests are ordered by their priority to optimize its data gathering responsive to the importance of a data gathering request. In a preferred embodiment, the priority of a crawl request is assigned by the policy engine, which examines the crawl request in light of priority-setting rules. The priority-setting rules are generated to assess the importance of a crawl, for example, whether the request is for an interactive user or for a background need, how many users or scheduled crawls are waiting for the data, or, in an embodiment in which priorities are reassigned to requests in the outgoing queue, a priority is reassigned responsive to the length of time the request has been waiting in the queue.

To service the crawl requests, a plurality of minions retrieve the crawl requests and crawl the target web sites to obtain the necessary data. A minion is a small executable program, such as an applet, that is specifically designed to retrieve collections of data. The small size of the minions enable them to be placed on any computer. Thus, in accordance with the present invention, minions are placed at numerous different servers and other sites that are physically near the different target site locations. Thus, because of the proximity of the minions, the crawl of a target site is accomplished quickly. Also, due to the number of minions, a target site can be accessed from multiple different pathways. Therefore, if one minion is unable to crawl a target site, because of congestion or network failures, a different minion located elsewhere in the Internet can still access the target site and provide the requested crawl. Thus, the present invention provides a fault-tolerant crawl as well as a minimal impact crawl.

The results of a crawl are transmitted to an incoming queue. In yet another embodiment of the present invention, multiple parsers are used to parse the retrieved data. The parsers are each preconfigured to retrieve a specific item or items from a collection of data. For example, in an auction embodiment, a parser may be designed to look only for the last bid price of an auction. Therefore, when a minion retrieves the data for the web page containing the auction, the preconfigured parser can quickly and accurately retrieve the requested data. The crawler identifies a preconfigured parser to perform a parse is identified at the time the request is generated, and upon the return of the minion, the crawler applies the identified preconfigured parser to the web page. The use of tailored parsers allows for fast fault identification and remediation. For example, if a target site happens to change its data layout for one of its fields, a traditional parser that parsed an entire page would fail when it encountered the changed field, and the system developer would be forced to examine the entire page to determine which field had changed. However, in accordance with the present invention, if a tailored parser designed to look for a specific field fails, the system developer knows immediately which field on the target site's web page has changed, and the system developer can modify the parser quickly. Other requests for other data on the page would be handled by different parser, and those requests would be satisfied. In contrast, in conventional systems with a single parser, if one field is altered, no requests for data on that page can be satisfied until the single parser is fixed. Thus, a system, method, and apparatus is provided that provides a fault-tolerant crawler that has a minimal impact on a crawled site.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a preferred embodiment of a crawler in accordance with the present invention.

Figure 2 is a flow chart illustrating a preferred method of operation of a client engine using a historical database.

Figure 3 is a block diagram of an LUID data structure.

Figure 4 is a block diagram of a data structure for the output queue.

Figure 5 is a flow chart illustrating a preferred method of operation of a policy engine.

Figure 6 is a graph of auction site usage.

Figure 7 is a block diagram of a network in which minions are dispersed.

DETAILED DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a preferred embodiment of a crawler 144 in accordance with the present invention. The crawler 144 is the software, hardware, or software/hardware combination that performs a crawl on a target site 128 in accordance with the present invention. A crawl is defined as the act of fetching a collection of data, typically a web page, from the target site 128, or crawlee. Although target sites are 128 are specifically described as web sites herein, the crawler 144 of the present invention enables crawling of almost any repository of organized of information, including the major commodity, securities and stock markets. A request source 136 generates a request for a specific item of data. A request source is any computer process that requests information from a data address (typically a URL – Universal Resource Locator, which is an address defining a path to a file or other data on the World Wide Web or other network or Internet facility). The request sources 136 typically access the crawler 144 through an Internet or other network connection. The connection may be made in accordance with a TCP/IP protocol or any other communications protocol that enables communication between two computer processes. Typically, a user will access the crawler's web site through the user's Internet connection, and then input a request for a specific item into a form provided by the crawler's web site. Alternatively, the request source 136 may be a computer application that is automated to search for new target sites. For example, in an auction embodiment, a request source 136 may be a computer application designed to discover new auction sites. Each request for data includes a URI that identifies a specific item of data from a collection of data. For example, a user may request the crawler 144 to track a particular auction for a camera that appears on an auction site. In this example, the URI would identify the web site (the collection of data) that hosts the auction, and the subdirectory at which the auction(s) for the camera (the specific item) are located.

A real time client engine 100 receives 200 the request for data. A preferred operation of the client engine 100 is illustrated in Figure 2 and the client engine 100 will discussed with reference to both Figures 1 and 2. In a preferred embodiment, the real time client engine 100 converts 204 the request information into an LUID (logical user id). An LUID is a data structure used to store and organize data used by the crawler 144 and to uniquely identify each auction monitored by the crawler 144 of the present invention. A preferred LUID data structure 300 is illustrated in Figure 3. An LUID 300 typically comprises URI information 304 and, in an auction embodiment, auction information 308. A URI is a Universal Resource Identifier, which is a text string which provides information concerning the location of a page. Specifically, an URI contains a protocol (usually "http"), a host name (such as www.auctionwatch.com) and a path within the machine (such as "/hello.html"). The LUID also has a unique identification number 312 used by the crawler 144 to internally differentiate different auctions, or other collections of data. The use of LUIDs 300 to track data stored on target sites 128 provides a high degree of flexibility to the crawler 144. For example, if a target site 128 changes its URI, the change is transparent to majority of the crawler components. Only the client engine 100 and the minions 124 (described below) must be able to recognize the new URI. The rest of the crawler 144 is unaffected by the change in URI.

The client engine 100 associates the LUID with a client engine identification and a parser identification. This allows the crawler 144 to identify which parser 108 to use when the collection of data is retrieved from the target site 128, and to which client engine to route the requested data. In a preferred embodiment, the real time client engine 100 identifies 208 a parser 108 that is preconfigured to parse retrieved collection of data for the specific item requested. The operation of the parsers 108 is described in more detail below. There is a preferably a one-to-one mapping of client engines 100 and request sources 136. Thus, the client engine 100 is preferably implemented as a plurality of daemons running on a UNIX/LINUX machine. Each daemon is spawned and cloned to connect with and handle user processes requesting data. A client engine 100 is typically a PHP (Personal Home Page (3)) module running in an Apache server process on a Linux server, although any web scripting language, such as Java Script or Perl, to create any type of server process on any server type (Linux, Unix), could be used in accordance with the present invention.

In a preferred embodiment, the client engine 100 checks 212 a historical database 104 to determine if the request for data can be satisfied by the data stored in the historical database 104. The historical database 104 is used to store web pages that have been previously retrieved by the crawler 144. As explained below, the crawler 144 retrieves web pages identified by requests received from request sources 136 and as part of scheduled crawls. The crawler 144 stores the web pages retrieved into the historical database 104. In this embodiment, the client engine 100, upon receiving a request, will examine the historical database 104 to determine 216 if the requested data is stored in the historical database 104. If the data is stored in the historical database 100, the client engine 100 can satisfy the request without initiating a crawl. In a preferred embodiment, the crawler 144 stores data into the database 104 along with a time stamp to indicate the time at which the data was retrieved. In this embodiment, the client engine 100 determines 220 whether stored data is valid. Stored data is classified as being valid if the stored data has been crawled within a predefined period of time. The period of time may be selected based on the nature of the data or the target site 128. For example, if a request is for a current event news story on a web site, the window may require that the data have been crawled within the last four hours. For data that changes less frequently, such as a description of an archived news story, the window may permit data to be retrieved from the database 104 if it has been crawled within the last seven days or longer. In an embodiment in which the crawler 144 verifies the validity of data in the database 104, the policy engine 116 preferably stores the policies governing when a site 128 should be crawled. For example, a policy for a target site 128 may be "crawl if the data is older than ten hours." In this example, the client engine 100 passes a request to the policy engine 116, as discussed below, which then checks a policy stored for the target site 128 and the date/time stamp of the data in the historical database 104 that would satisfy the request to determine whether the data in the database 104 may be used to satisfy the request.

If the data is invalid, or the data is nonresponsive, the client engine 100 passes 218 the LUID 300 to the policy engine 116 to generate policies for the request for data. If the historical database 104 is able to satisfy the request for data, a parser 108 is assigned (as discussed below) and parses 224 the web page or other collection of data to obtain the specific item requested. The requested information is then transmitted 228 back to request source 136, without ever requiring a crawl of the target site 128. Thus, the use of the historical database 104 minimizes the number of crawls required to be performed by a crawler 144. A further benefit of implementing a historical database 104 is that it provides a greater degree of reliability to users attempting to access information about a target site 128 that has been crawled by the crawler 144. If the target site 128 is unavailable to the public, the crawler 144 is still able to satisfy requests with the stored data in the historical database 104. Moreover, if servers at the target site's main address are not accessible, for example, as a result of a third party attacking a target site 128 by sending an overly burdensome number of requests to the site 128, the crawler 144 of the present invention is still able to satisfy requests because the exact URL for a given web page is stored in the database 104. This allows the crawler 144 to bypass the main path to the target site 128 and access the target site's direct address for the item of interest. Thus, the crawler 144 will provide connectivity to a target site 128 even in circumstances when main path of access to the target site 128 is malfunctioning. This is beneficial to both the crawler 144 and the target site 128, as the target site 128 is able to maintain a revenue stream even during periods of partial system failure, and the crawler 144 is able to provide its services regardless of these problems with the target site 128.

In an auction monitoring embodiment of the present invention, auction pages are stored in the historical database 104. In a typical auction page, the only changing information over time is the price, and occasionally, the closing time or date. The description of the product, the seller, the auction host, and other information remains the same. Thus, if a request for data is for any of these constant data fields, the cached web page of the auction would satisfy the request. Thus, upon receiving a request for information from an auction web site page, the client engine 100 checks the historical database 104 for the web site page, requests a parser 108 designed to search for the specific information requested, and returns the information to the request source 136. If the request is for the price or other changing information, the request is passed to the policy engine 116. Once data is retrieved from a target auction site 128 to satisfy a request, the appropriate field in the historical database 104 is updated, and a subsequent request may not require a new crawl. Thus, by using a historical database for an auction embodiment of the present invention, the crawler 144 minimizes web traffic on an auction site.

A preferred operation of the crawler policy engine 116 is illustrated in Figure 5, and the policy engine 116 will be described below in connection with Figure 1 and Figure 5. First, the crawler policy engine 116 receives 500 an LUID 300 from the client services engine 100 to identify 504 the target site 128. Alternatively, the policy engine 116 can receive LUIDs 300 from a scheduler 112. The scheduler 112, as discussed below, generates scheduled crawls that are designed to periodically update target site data or discover new target sites, and provides automatically rescheduling of crawls, as discussed below. Then, in a preferred embodiment, the policy engine 116 builds 508 a policy or set of policies for the target site 128. A policy is a rule or set of rules, typically implemented as executable code such as in Java Script, governing the crawling of a target site 128. The policies are generated by an operator of the crawler 144 based on data received from the target site 128 or general understanding of the industry. The policies are preferably stored in separate databases organized by different topics, and the policy engine 116 builds policies for a crawl upon receiving an LUID 300 by examining all of the databases maintained by the policy engine 116 and applying those policies relevant to the particular target site 128, as identified by the LUID 300.

For example, a database may be created for time of day, or target site, or connectivity loss, or site closure. Thus, when the policy engine 116 receives an LUID 300, it will examine the time of day database to determine if a policy is applicable to the crawl request based on the current time of day. The policy engine 116 will examine all of the databases to build a list of all policies that apply to a particular request for data. Storing policies in different databases increases the flexibility of adding and deleting policies. For example, if a system operator wanted to add a policy that no crawls should be conducted if they are received from a specific user, then the system operator can simply add the user's ID to an existing prohibited user database. Once a prohibition on a user was listed, then the user could be removed from the database. Thus, the use of a plurality of databases to organize the policies provides a fast and efficient mechanism for applying and deleting policies.

The use of crawling policies for each target web site 128 allows for a more efficient and more harmonious crawling application. Crawls can be tailored for each web site's capabilities, and therefore the crawls will be more efficient. Crawls can be scheduled to accommodate web site timing preferences, and thus generates a more harmonious relationship between the crawler 144 and the target site 128. Policies can be as complex or as simple as needed to provide the most efficient crawl. An exemplary set of policies includes:

- Do not crawl if web site is shut down
- Only crawl if latency of connection is less than a predetermined value
- Crawl only on off-peak times
- Crawl anytime (for a "friendly" web site)
- Never crawl
- Crawl only after a predetermined time period has elapsed.

After the policy engine 116 has retrieved all potentially applicable policies, the policy engine 116 applies the policies to determine 512 whether a crawl should occur. If the crawl is not permissible at this time according to one of the policies, a crawl request 400 (discussed below) is stored 524 in a request queue maintained by the policy engine 116 for later transmittal or sent to the scheduler 112 for rescheduling, if the crawl request 400 is from a scheduled crawl. If a crawl request 400 is rescheduled by the scheduler 112, upon being received again by the policy engine 116, the policy engine 116 builds new policies for the rescheduled request to account for any new policies that may apply to the crawl request 400.

A crawl request 400, as shown in Figure 4, is a data structure used to store information used to perform the crawl, and to analyze and route the crawl results once obtained. A preferred crawl request data structure comprises a URL (or other identifier) 408, a parser identification 404 to identify the parser 108 that will be parsing the retrieved data, and a client identification 312 that identifies the client engines 100 to which the parsed information will be routed. In the simplest case, one client engine 100 will desire a single URL to be parsed by one specific parser 108, and therefore the crawl request 400 will comprise only one URL field 408, one client ID field 312, and one parser ID field 404. In a more complex case, two client engines 100 (or client engines 100 and any scheduled automatic crawls) could desire the same URL, but desire different information from the URL. Accordingly, prior to placing a request for data in the queue 120, the policy engine 116 checks the requests for data currently in the queue 120 to determine if any request for data in the queue 120 is for the same target site 128. If there are, the policy engine 116 combines the multiple requests 400 into a single crawl request 400 comprising one URL field 408, and two client and parser ID fields 312, 404. The two client ID fields 312 identify the two request sources 324 (or the scheduler 112 in the case of an automated crawl), and the two parsers ID fields 404 identify the two parsers 108 preconfigured to parse the returned web page or other collection of data for the specific item requested. Thus, by coalescing multiple requests from different request sources 136 for similar data into a single crawl, the crawler 144 of the present invention provides lower overhead, lower latency and globally higher throughput. It also decreases the crawling burden on the target site 128.

If the crawler 144 determines that a crawl should occur, in one embodiment, the policy engine 116 determines 516 a priority for the crawl request 400 based upon a second set of criteria including:

Whether the request is for an interactive user or for a background need (as indicated by the source identification associated with the LUID 300)

How recently was this URL crawled

When is the URL scheduled to be crawled next

How many users are waiting for the data

How long the request 400 has been waiting in the queue 120

Time of day (off-peak times are better for crawling)

The second set of criteria establishes priorities for the requests 400 to be placed in the outgoing queue 120. In one embodiment, priorities are assigned to crawl requests 400 through the use of an importance scale (for example, a scale of 1 to 10, 1 being process immediately and the others giving hierarchical less importance to a crawl request 400). The actual scale can be implemented in numerous ways and the implementation is not critical to the operation of this aspect of the present invention). In a further embodiment, the priority system is implemented using fuzzy logic as is known in the art, and thus relative priority levels are assigned to the different crawl requests 400. Once the priority of a crawl request 400 is established, the crawl request 400 is sent 520 to the outgoing queue 120, and the crawl requests 400 are retrieved from the outgoing queue 120 responsive to their priorities. Thus, in this embodiment, the policy engine 116 decides not only whether a request may be transmitted at that time for a given target site 128 but also decides the priority between competing requests for different target sites 128.

For example, if a first request to a target site 128 is determined to be appropriate for transmission, and is a user-demand request, and a second request to a different target site 128 is also determined to be acceptable but is from an automated discovery crawl, the user-demand request is given a higher priority than the request from the automated discovery crawl. The use of this dynamic priority setting greatly improves the efficiency and responsiveness of the crawler 144. In another example of the application of a different priority rule, if a request would have been given low priority, but the policy engine 116 determines that a number of other requests for the same target site 128 are waiting in the queue 120, the policy engine 116 will increase the priority of the request. Thus, the 'popularity' of a request can increase its priority, and the priority of a request can be dynamically reassigned even after being placed in the queue 120. The ability to dynamically set priorities in queues for a crawler 116 allows the most important or most requested information to be retrieved first, and secondary information to be retrieve in the ordinary course of operations.

A crawl request 400 stays in the queue 120 until the crawl request 400 is the highest priority request waiting in the queue. Then, a minion 124 will pull the request from the queue 120 and fetch the page from the target site 128 that contains the requested data. In a preferred embodiment, once a request is fed to a minion 124, the output queue 120 automatically schedules a second crawl request 400 to be made in the near future on the expectation that the first crawl request 400 will fail. If the first crawl request 400 is successful, then upon return of the data to the incoming queue 132, the second crawl request 400 is canceled. However, by providing automatic rescheduling, the crawler 144 of the present invention is very tolerant to any socket errors present in the network. Moreover, automatic rescheduling also helps to lessen traffic on a target site 128. For example, if a minion 128 fails, there is a delay in retrieving data as a second crawl must be performed in accordance with the automatic rescheduling. However, due to the delay, a user may request a second crawl (e.g., by hitting a "Reload" button on a browser). This second crawl will <u>not</u> be executed by the crawler 144 because the automatically scheduled second crawl will have retrieved the data and stored the results in the database 104, which will then be used to satisfy the second crawl. The second crawl will not be executed.

Although the policy engine 116 has been described as building policies, applying policies to determine if a request is appropriate at that time for the target site 128, applying a second set of rules to determine the priority of a request, and transmitting the requests either to the scheduler 112 or the output queue 120, all of the above functionality is not required in accordance with the present invention; any part or subset of this functionality could also be used and provide benefits as described herein. Additionally, by combining an embodiment of the present invention in which the crawler 144 coalesces multiple requests in the outgoing queue 120 as described above with an embodiment using a historical database 104, the crawler 144 can provide information to a large number of users while making a minimal number of queries to a target auction site 128. Furthermore, if a portion of users shift from directly communicating with the target site 128 to communicating with a crawler 144, the traffic load on the target site 128 will be decreased. Finally, the application of the present invention will allow more users to access the target site's information than would otherwise visit the auction site 128, thus providing an additional benefit to the auction site 128.

In a further embodiment of the present invention, a scheduling engine 112 provides another source of requests for data. The scheduling engine 112 schedules regular crawls on the Web to discover new sites of interest and update information on existing site. In accordance with the present invention, the scheduling engine 112 is designed to minimize the impact of a crawl on a target site 128. For example, in an auction embodiment of the present invention, the scheduling engine 112 schedules crawls only for the beginning of an auction and after the auction is closed. Without additional demand, the crawler 144 will only perform these two crawls for a given auction. As shown in Figure 6, the crawler's first and final crawls take place at off-peak times. Therefore, the loading for the target site 128 is minimized due to the scheduling of the crawls at the times when the information is known to change. Furthermore, the demand driven crawls which do take place at "peak time" correspond to situations where the crawler 144 is likely to be able to satisfy several users with a single crawl and therefore reducing the load of the target site 128. Further, storing already crawled auctions in the historical database 104 as described above makes discovery of new auctions easier. Specifically, the crawler 144 does not have to recrawl an entire target site 128 to discover new auctions because the crawler 144 can examine the history database 104 to identify recently crawled portions of the target site 128, and schedule automatic crawls that bypass those recently crawled portions until such time as there is an explicit need for the information. Thus, in accordance with the described embodiment of the present invention, the scheduling engine 112 ensures that a target site 128 is crawled no more often than is required to meet the information needs of the crawler 144 and its request sources 136, thereby eliminating unplanned crawling.

In a preferred embodiment, and as illustrated in Figure 7, a plurality of minions 124 are employed to fetch data. A minion 124 is a component of the crawler 144 that resides on a (potentially) remote computer and fetches URLs under control of the crawler 144. In implementation, a minion 124 is an executing program, for example a daemon written in C, Perl, or any other language. Minions 124 are preferably implemented to execute on servers 700 or other computing devices that reside anywhere. Minions can work as background daemons on UNIX or LINUX machines or as tasks on Windows-based personal computers, or on personal digital assistants. Minions can also run as JAVA Applets anywhere on the network. A minion 124 can talk to the Internet via an OC3 connection or a dial up modem, or the like. In a preferred embodiment, a minion 124 runs as a daemon on a server 700 co-located at the target site's machine room 704 and performs the crawl by bypassing the web interface; however, in general, minions 124 are located topologically as close as possible to the target site's server 708. By locating the minions 128 topologically close to the servers of the target sites 128, the crawler 144 of the present invention provides a fast and efficient crawl. Due to the small size of the minions 124, the minions 124 can be stored in a very small computing environments. Therefore, it is easy to find a location near the target site's server to which to store a minion 124.

The use of minions 124 provides a fault tolerant crawl. The size of the minions 124 allows many of them to be used to retrieve data. By placing the minions 124 on as many different backbones and different locations as possible, as shown in Figure 7, the minions 124 can approach target site server 708 from a sufficient number of different paths to avoid network failures and avoid or work around bottlenecks in network communications that would ordinarily prevent a crawl. For example, if the Atlantic backbone was experiencing transmission difficulties, a crawl in accordance with the present invention could still reach European servers 700 because some minions 124 would attempt to access the servers 700 from the Pacific backbone. Thus, the use of a plurality of minions 124 collocated in different areas in the world provides a crawl that is highly resistant to adverse network and environmental conditions. The loss of any subset of minions 124 will result in an instantaneous decrease in the number of URLs which can be fetched, but no loss of data since the URLs are already scheduled to be crawled a second time (should the second attempt to crawl fail, the third attempt has already been scheduled, and so forth.) The loss of connectivity to/from any major region of the internet is similarly avoided as it would be viewed as a loss of the minions 124 in that region and consequent automatic rescheduling to different minions 124 located elsewhere. In fact, there could be a total loss of connectivity between the target site 128 and the crawler subsystem 140 and yet the crawler 144 can still operate at full capacity. Thus, by using a widely disbursed cloud of minions 124, a crawler 144 is able to reduce the possibility that there will be no communications path between the crawler 144 and the target site 128. If the number or minions 124 sufficiently large, disbursed and constantly moving, the possibility of total communications failure becomes remote.

In one embodiment a minion management module spawns and manages minions 124. In this embodiment, a daemon is employed to monitor the rate of service and to decide whether to increase the pool of minions 124 or to slow the rate of requests. The daemon may base a minion adjustment decision on factors such as when the number of requests in the queue 120 exceed a threshold, when the response time (or some calculated average time) from a target site 128 to a request exceeds a threshold, and/or when low priority requests are only being processed at off-peak times. These factors are designed to characterize a situation in which more minions 124 can profitably be spawned to increase the efficiency of the crawler 144. Other factors to perform similar functions could also be used to determine when to spawn additional minions in accordance with the present invention.

In a further embodiment, minions 124 are tailored to retrieve specific data from target sites 128. For example, if a particular auction site 128 is to be crawled, a minion 124 may be tailored to crawl only for a specific item, in accordance with the target's data storage format, server operating system, bandwidth, etc. By providing unique minions 124 to crawl web sites for which they are custom designed, the crawls can be much more efficient than a generic crawl, thus minimizing the impact of the crawl on the target web site. In a further embodiment, a minion 124 may be instructed to use cookies or ignore them based upon the categorization policy to make the crawler's data gathering as benign as possible.

In a further embodiment, a minion 124 is designed to embody a request for a specific item to emulate a human's request for items. Emulating human behavior allows a minion 128 to take advantage of built-in optimization (such as caching) techniques already present in a network. Thus, certain types of requests may be transmitted to a specific minion 128. For example, a specific minion 124 may crawl any site, but when crawling, the minion 124 may be restricted to a particular area (such as "stamps & coins") so that the behavior of the minion 124 will emulate human behavior. Furthermore, a given minion 124 may be kept dormant for a random period of time to reduce its average bandwidth demand. This also helps emulate human behavior and helps avoid dominating resources on the target site 128.

There is no practical limit to the number of minions 124 to use; in one embodiment, the crawler 144 collocates servers throughout the world, and minions 124 are stored on the collocated servers. In another embodiment, minions 124 use accounts at different Internet Service Providers. Minions 124 may also be implemented to be multithreaded and/or use multiple IP addresses and/or use proxies (such as Squid). Minions 124 also do not need to have live socket connections to the crawler 144. In one embodiment, to address non-real time queries, the crawler 144 uses a batch facility (such as email) as a distributed queuing mechanism. The minion 124 can then email the results of the crawl back to the crawler subsystem 140. In this embodiment, a minion 124 performs a crawl response to an e-mail message containing one or more crawl requests from the scheduler 112. Additionally, minions 124 may have asynchronous bandwidth availability. Common connections such as 56K dial up (V.90) modems and ADSL connections provide higher download bandwidth than upload bandwidth. To address this mismatch, in one embodiment, the minions 124 use local CPU cycles to compress the crawled information before sending it to the crawler subsystem 140.

In a further embodiment, minions 124 may be used to crawl sites 128 that may require authorization from the target site 128. In this embodiment, a minion 124 is designed to provided the authorization required by the target site 128 for crawling the site 128, allowing the target site 128 to selectively decide whether and to whom it wants its site 128 crawled. For example, a company may host an Intranet in which employees are permitted to auction goods. In this embodiment, the company may contract with the operators of the crawler 144 to allow the crawler 144 to design a minion 124 to access their servers through their firewalls to allow the auction be accessed by the public, while maintaining their security for their other data.

Once a minion 124 retrieves data (usually the web page), it stores the results in an incoming queue 132. The results, along with a timestamp, the URL and all client/parser pairs, remain in the incoming queue 132 until the results have been processed by all requested parsers 108 (and the corresponding results fed to all client engines 100 and then to the request sources 136). The incoming queue 132 also deletes any automatically rescheduled requests from the outgoing queue 120 after receiving a successful retrieve of data.

In a preferred embodiment, multiple parsers 108 are used to parse the data retrieved by a minion 124. The parser 108 to perform the searching is identified by the client engine 100, as discussed above, and the parser ID is placed in the routing tag that accompanies the crawl request 400 in the outgoing queue 120. Multiple parsers 108 are used to provide fast, specific, and efficient indexing and retrieval of data. For example, in one embodiment, a different parser 108 is used for each different data format of target web sites 128. Moreover, in another embodiment, different parsers 108 are used for each different data fields of a web page. For example, one parser 108 is designed to locate the price of an auction that is hosted by a specific web site. This parser 108 is able to very quickly locate and retrieve the information because it knows exactly where to look on the page. A crawler 144 using multiple parsers 108 is fault tolerant because if a specific parser fails to retrieve its data, for example, if the web site has changed its data format, the programmer can immediately rectify the error because the programmer knows exactly which field to look for. For example, if a parser 108 designed to retrieve closing prices of a stock fails, the programmer can go to the closing price field of the retrieved data and determine what has changed. In conventional systems, the parser crawls the entire page, and if it fails, the programmer will not know which field has changed without examining the entire page. However, in accordance with the present invention, the programmer will immediately know which field has changed because the programmer will know which parser 108 has failed. In addition, if one parser fails, the incoming queue 132 will only back up for requests requiring that specific data field. If there are requests for other components of the web page, those requests can be serviced by the other parsers 108.

In one embodiment, one parser 108 is designed to store an unmodified page into the historical database 104 (preferably in compressed or distilled form). This image is stored as an URL with a time stamp. Then, the crawler 144 can use the stored page image for short term reparsing. In the embodiment discussed above, the parsers 108 parse only for specific items, and thus if a request is generated for the request source for the same item, the crawler 144 may satisfy the request from the information stored in the historical database 104. However, in an embodiment in which the unmodified page is stored, if the request is for an item stored on the page that had not been previously requested, the crawler 144 can check the historical database 104 and determine if the stored page is in the database 104. If it is, the crawler 144 can assign the appropriate parser 108 to parse the page to retrieve the specific item. Thus, this embodiment provides a further method for eliminating crawls. Each time this page is reparsed, it has saved one complete crawl on the target site 128.

Additionally, data mining applications will find the original image of the page valuable since it represents the maximum possible information content. To support this function, auxiliary information such as pictures and descriptive text which reside on associated URLs are saved in the historic database 104 as well. Finally, when the target site 128 changes its URL contents, including URL format changes, page layout changes or other changes which results in a failure of the existing parsers 108 to gather information, the crawler operator can use the stored page in the historical database 104 to modify a parser 108 to adjust to the new URL information without requiring a new crawl.

Although the present invention has been described in a web-based embodiment, the minions 124 and parsers 108 can be modified to operate on any type of information and provide the benefits described above. Thus, additional minions 124 and parsers 108 can be added to crawl non-URL based information, including stock information (identified by ticker symbol), commodity information, currency transactions, and the like. To enable the minions 124 and parsers of the present invention to crawl and parse the target site 128, the target site 124 must its information with a unique identifier, and the data must be accessible to a specialized minion to fetch the information. If those two conditions exist, then the crawler 144 of the present invention can be applied to provide minimal impact, yet highly efficient and flexible, data gathering on any target site 128.

The crawler 144 of the present invention is capable of growth an order of magnitude in capacity within a short period of time through the dynamic addition of resources where needed. For example, as demand grows more minions 124 and more parsers 108 can be added to increase the processing power of the crawler 144. In contrast, conventional systems must replace and upgrade components in order to provide higher processing capability. Further, in a multithreaded embodiment, the crawler 144 of the present invention is capable of crawling millions of URLs each hour by implementing the different components of the crawler 144 in parallel.

The foregoing describes in details the features and benefits of the present in various embodiments. Those of skill in the art will appreciate that present invention is capable of various other implementations that operate in accordance with the foregoing principles and teachings. For example, the arrangement and organization of the various functionalities, such as generating and converting an LUID 300, checking the historical database, scheduling, routing information, can be performed by different components of the crawler 144 as would be known to those of ordinary skill in the art. Certainly, the names of the various entities may be changed without impacting their functional operations. Accordingly, this detailed description is not intended to limit the scope of the present invention, which is to be understood by reference the claims below.

CLAIMS

1. A system for providing a minimal impact crawl, comprising:
   a client engine, coupled to receive a request for data to be accessed from a target site;
   a policy engine, coupled to the client engine to receive the request for data, maintaining crawling policies for sites to be crawled, and to apply a policy associated with the target site to determine scheduling of a crawl to gather data from the target site.

2. The system of claim 1 wherein requests for data are converted into crawl requests, the system further comprising:
   an output queue, coupled to the policy engine, for storing crawl requests in an organization responsive to a priority assigned to the crawl request; and wherein the policy engine assigns priorities to crawl requests responsive to the application of policies associated with the target site of a crawl request.

3. The system of claim 2 further comprising:
   a plurality of minions, at least one of which is located topologically near the target site, for retrieving a crawl request from the output queue, and crawling the target site to gather the requested data.

4. The system of claim 3 in which a minion retrieves a collection of data from a target site further comprising:
   an incoming queue, for receiving crawled data from at least one minion; and
   a plurality of parsing engines, coupled to the incoming queue, wherein each parser is preconfigured to retrieve specific data from a retrieved collection of data.

5. The system of claim 4 further comprising:
   a plurality of client engines coupled to receive requests for a specific item of data, coupled to the policy engine to transfer the requests for data to the policy engine, and for selecting a parser to parse retrieved data that is preconfigured to parse for the specific item identified in the received request.

6. The system of claim 2 further comprising:
   an incoming queue, for receiving crawled data from at least one data gathering engine; and
   a plurality of parsing engines, coupled to the incoming queue, wherein each parser is preconfigured to retrieve specific data from a retrieved collection of data.

7. The system of claim 6 further comprising:
   a historical database, for storing results of a previous crawl; and wherein
   the client engine is coupled to receive a request from the request source, parses the request to identify an address at which the requested data can be located, accesses the historical database to determine whether the data requested is stored within the historical database, and, responsive to determining data requested is stored within the historical database, retrieves the data and transmits the retrieved data to the request source to avoid crawling the target site,

8. The system of claim 7 wherein the parsing engines transfer the retrieved data to the historical database; and wherein the request engine transfers a request for data to the crawling engine responsive to the data requested not being stored within the historical database.

9. The system of claim 8 wherein the parsing engine stores date and time information indicating the date and time the crawl was performed to retrieve data into the historical database, and request engine retrieves data from the historical database responsive to whether the stored data has been crawled within a predefined window of time.

10. The system of claim 1 further comprising:
   a scheduling engine, coupled to the policy engine, for scheduling crawls responsive to receiving target site information and time and date information, and wherein the policy engine provides target site information to the scheduling engine and provides time and date information to the scheduler.

11. The system of claim 10 wherein the policy engine determines the time and date information responsive to the request for data and the policy engine's application of policies associated with the target site.

12. The system of claim 10 wherein auction sites are crawled, and the scheduling engine automatically schedules crawls for a beginning and end of an auction.

13. The system of claim 1 wherein policies include prohibitions on crawling at specified times or dates.

14. The system of claim 1 wherein policies include prohibitions on crawling from a specified request source.

15. The system of claim 1 further comprising:

a historical database, for storing results of a previous crawl; and

the client engine parses the request to identify an address at which the requested data can be located, accesses the historical database to determine whether the data requested is stored within the historical database, and, responsive to determining data requested is stored within the historical database, retrieves the data and transmits the retrieved data to the request source to avoid crawling the target site.

16. A system for gathering data from an electronic database connected to a network, comprising:

a request engine, coupled to receive a request for data to be accessed from a target site; and

a plurality of minions, at least one of which is located near the target site, for retrieving the request for data from the request engine, accessing the target site to gather the requested data, and for transferring the requested data to the request engine.

17. The system of claim 16 wherein a minion is designed to retrieve specific data from a predesignated target site, and general data from at least one other target site.

18. The system of claim 16 wherein a minion is located in a machine room hosting a server of the target site.

19. The system of claim 16 wherein a first minion is coupled to a server of the target site along a first pathway, and a second minion is coupled to the server of the target site along a second pathway.

20. The system of claim 16 wherein at least two minions have different internet protocol addresses.

21. A system for gathering data from an electronic database connected to a network, comprising:

at least one client engine, coupled to receive a request for specific data from a collection of data;

a data gathering engine, coupled to the client engine, for gathering the collection of data related to the specific data requested; and

a plurality of parsing engines, coupled to the data gathering engine and the client engine, each parser preconfigured to retrieve specific data from a retrieved collection of data and transfer the retrieved specific data to a client engine requesting the specific data.

22. A system for minimizing the impact of a crawl on a web site, comprising:

a historical database, for storing results of a previous crawl; and

a request engine, coupled to receive a request from a request source for data to be accessed from a target site, parsing the request to identify an address at which the requested data can be located, accessing the historical database to determine whether the data requested is stored within the historical database, and, responsive to determining data requested is stored within the historical database, retrieving the data and transmitting the retrieved data to the request source to avoid crawling the target site.

23. The system of claim 22 further comprising:

a crawling engine, coupled to the request engine, for retrieving data from target sites responsive to receiving a request for data and transferring the retrieved data to the historical database; and wherein the request engine transfers a request for data to the crawling engine responsive to the data requested not being stored within the historical database.

24. The system of claim 23 wherein the crawling engine stores date and time information indicating the date and time the crawl was performed to retrieve data into the historical database, and request engine retrieves data from the historical database responsive to whether the stored data has been crawled within a predefined window of time.

25. A method of providing a minimal impact crawl:

receiving a request from a request source for data to be accessed from a target site;

applying a policy designed to provide a minimal impact crawl to the request for data to determine scheduling of a crawl of the target site; and

crawling the target site responsive to the determined schedule.

26. The method of claim 25 wherein a plurality of policies are stored, the method further comprising:

building a set of policies to apply to the request for data.

27. The method of claim 25 further comprising:

assigning a priority to a request for data responsive to the application of a policy associated with the target site of a request for data, wherein the priority determines an order in which requests for data are processed.

28. The method of claim 25 further comprising:

responsive to receiving a request for data, checking a historical database to determine if the data requested is present in the historical database; and

responsive to determining that the data requested is present in the historical database, transmitting the requested data to the request source to avoid a crawl of the target site.

29. The method of claim 25 wherein the historical database stores date and time information for data stored in the historical database, the method further comprising:

responsive to determining that the data requested is present in the historical database, determining whether the data has been stored within a predefined window, and transmitting the requested data further comprises transmitting the requested data responsive to determining the data has been stored within a predefined window.

30. The method of claim 25 further comprising:
storing scheduled crawls in an output queue;
examining the scheduled crawls in the output queue to determine if any scheduled crawls are designed to retrieve data from an identical address; and
responsive to determining that more than one scheduled crawl is designed to retrieve data from an identical address, replacing the more than one scheduled crawl to the identical address with a single crawl for the address.

31. A method of crawling a target site comprising:
receiving a request from a request source for specific data from a collection of data;
gathering the collection of data related to the specific data requested;
parsing the collection of data with a preconfigured parser designed to retrieve specific data from a retrieved collection of data; and
transfer the retrieved specific data to the request source.

32. A method of providing a minimal impact crawl:
receiving a request from a request source for data to be accessed from a target site;
checking a historical database to determine if the data requested is present in the historical database;
responsive to determining that the data requested is present in the historical database, transmitting the requested data to the request source to avoid a crawl of the target site; and
responsive to determining that the data requested is not present in the historical database, crawling the target site responsive to retrieve the requested data.

33. The method of claim 32 wherein the historical database stores date and time information for data stored in the historical database, the method further comprising:

responsive to determining that the data requested is present in the historical database, determining whether the data has been stored within a predefined window, and transmitting the requested data further comprises transmitting the requested data responsive to determining the data has been stored within a predefined window.

34. A method for providing a minimal impact crawl of an auction site comprising:
scheduling a first automatic crawl of the auction site at a beginning of an auction;
scheduling a second automatic crawl of the auction site at a time after the auction has ended; and
crawling the auction site in accordance with the schedule.

35. The method of claim 34 further comprising:
responsive to receiving a user request for information on the auction site, crawling the auction site for the requested information.

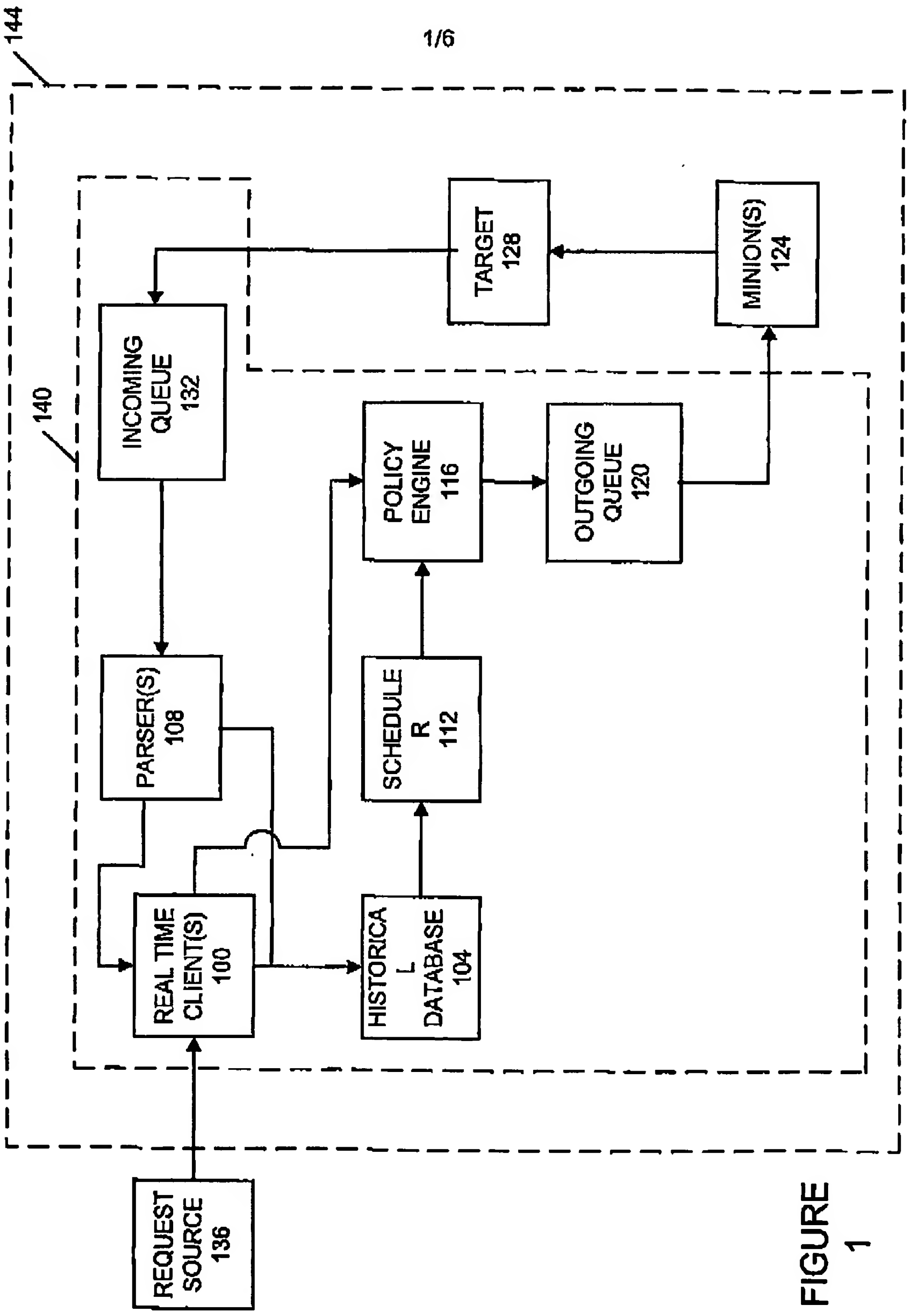

144
140
INCOMING QUEUE 132
TARGET 128
MINION(S) 124
POLICY ENGINE 116
OUTGOING QUEUE 120
PARSER(S) 108
SCHEDULE R 112
REAL TIME CLIENT(S) 100
HISTORICA L DATABASE 104
REQUEST SOURCE 136

FIGURE 1

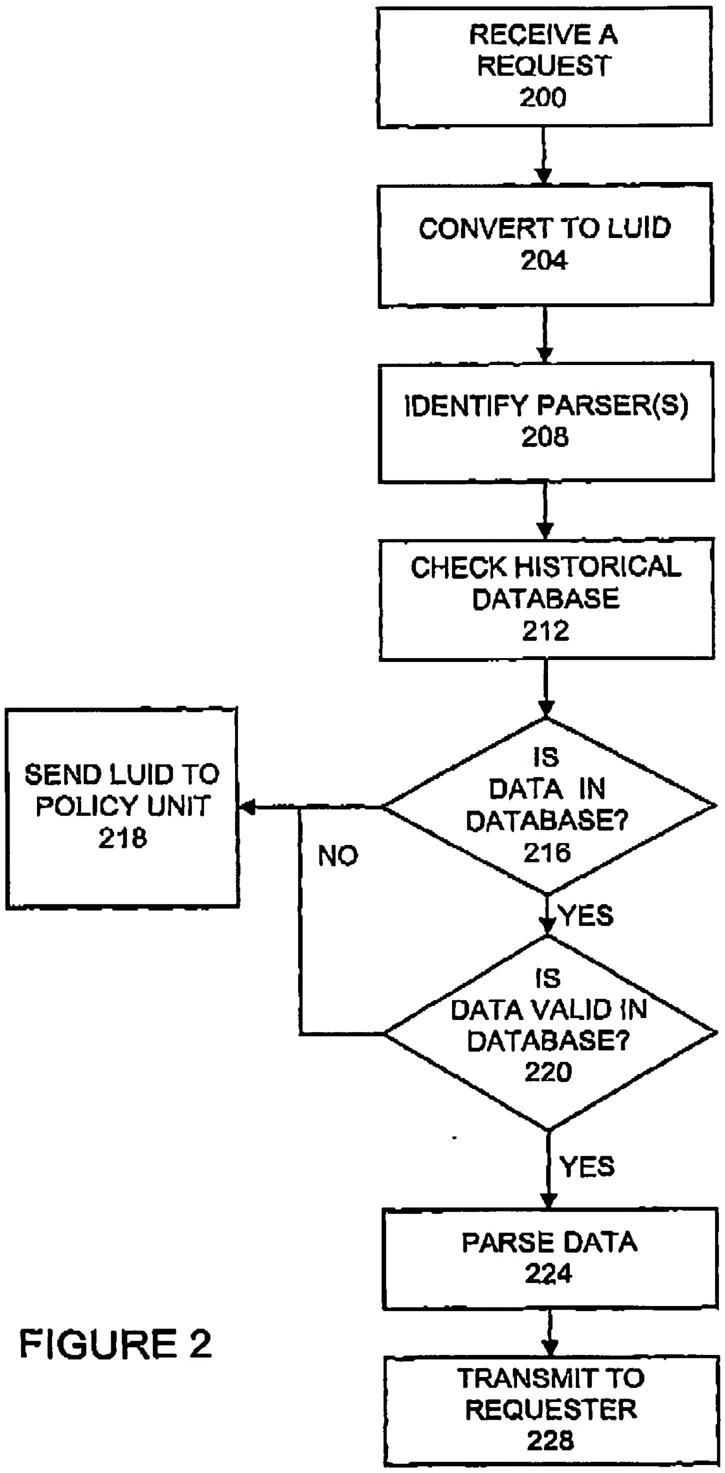

RECEIVE A REQUEST 200
CONVERT TO LUID 204
IDENTIFY PARSER(S) 208
CHECK HISTORICAL DATABASE 212
IS DATA IN DATABASE? 216
NO
YES
SEND LUID TO POLICY UNIT 218
IS DATA VALID IN DATABASE? 220
YES
PARSE DATA 224
TRANSMIT TO REQUESTER 228

FIGURE 2

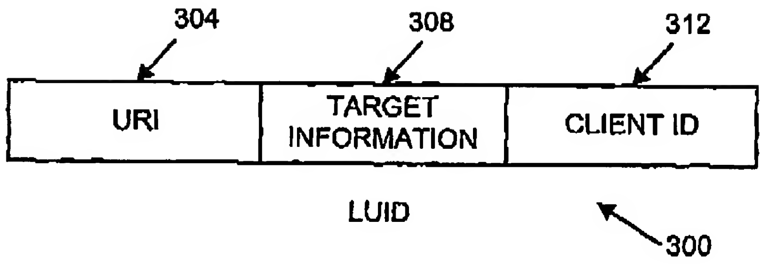


FIGURE 3

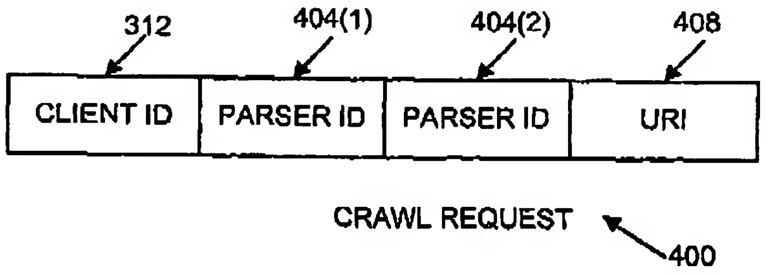


FIGURE 4

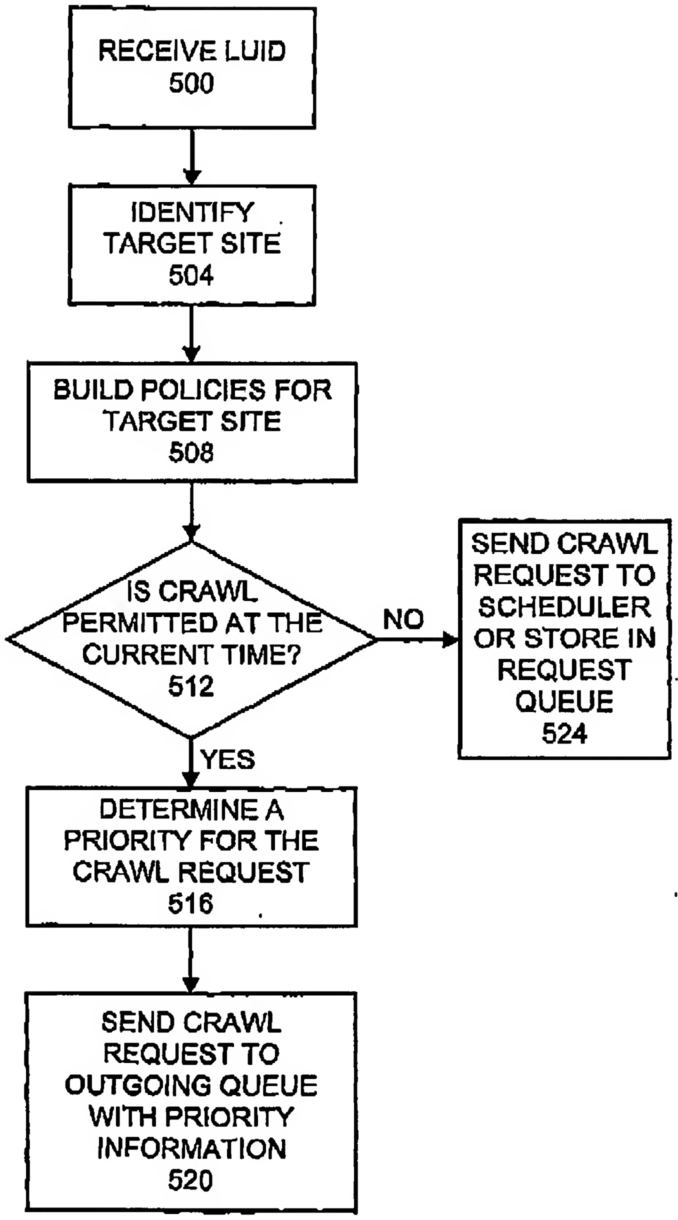

RECEIVE LUID 500
IDENTIFY TARGET SITE 504
BUILD POLICIES FOR TARGET SITE 508
IS CRAWL PERMITTED AT THE CURRENT TIME? 512
NO
SEND CRAWL REQUEST TO SCHEDULER OR STORE IN REQUEST QUEUE 524
YES
DETERMINE A PRIORITY FOR THE CRAWL REQUEST 516
SEND CRAWL REQUEST TO OUTGOING QUEUE WITH PRIORITY INFORMATION 520

FIGURE 5

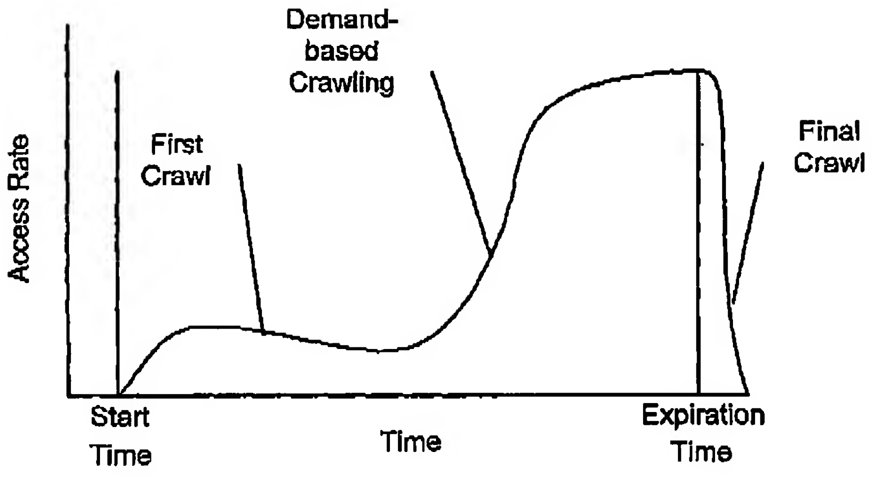

Access Rate
First Crawl
Demand-based Crawling
Final Crawl
Start Time
Time
Expiration Time

Figure 6

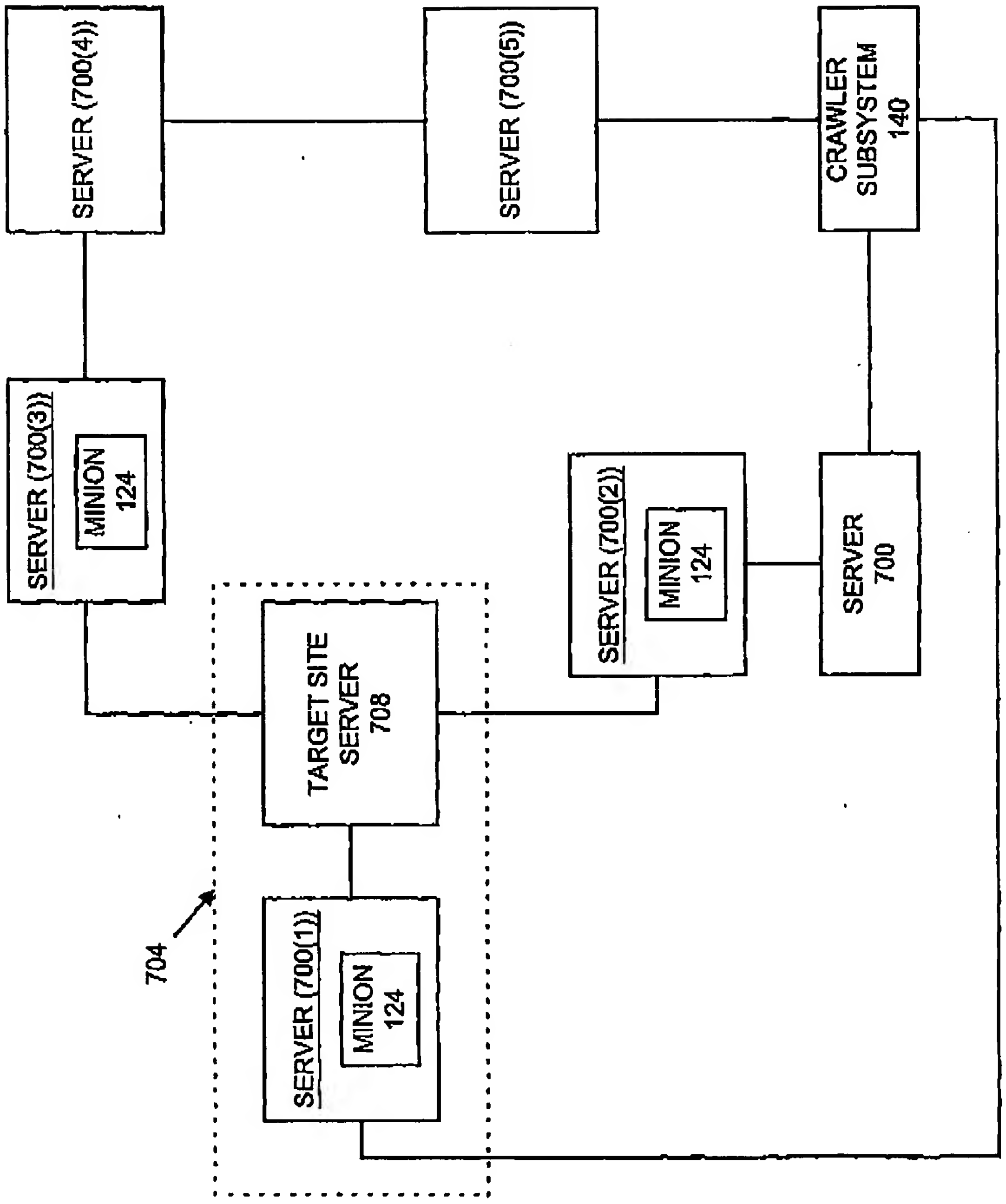

SERVER (700(4))
SERVER (700(5))
CRAWLER SUBSYSTEM 140
SERVER (700(3))
MINION 124
SERVER (700(2))
MINION 124
SERVER 700
TARGET SITE SERVER 708
704
SERVER (700(1))
MINION 124

FIGURE 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/35169

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) :G06F 17/00; G06F 17/30
US CL :707/1, 3, 4, 103, 104; 705/26, 27.
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 707/1, 3, 4, 103, 104; 705/26, 27.

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WEST
Search terms: web crawler, internet crawler, web sites, data, queue, parse, historical database, engine, minion, auction site.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,875,446 A (BROWN et al) 23 February 1999, col. 3, lines 1-58, col. 4, lines 9-40 and lines 57-63, col. 6, lines 12-46, col. 7, lines 22-67, col. 8, lines 1-22 and lines 58-67, col. 9, lines 1-67, col. 11, lines 33-55 and lines 65-67, col. 12, lines 1-11 and lines 63-67, col. 13, lines 1-9, col. 14, lines 35-55, and 16, lines 14-30.	1-35
Y	US 5,974,455 A (MONIER) 26 October 1999, col. 1, lines 16-67, col. 2, lines 1-67, col. 3, lines 3-17 and lines 29-64, col. 8, lines 53-61, col. 9, lines 10-18 and lines 28-62, col. 11, lines 44-67, col. 12, lines 1-3, col. 13, lines 17-67, col. 14, lines 1-67, col. 15, lines 10-23, and col. 16, lines 1-21.	1-35

[✓ X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance		
"E"	earlier document published on or after the international filing date	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search 23 FEBRUARY 2001	Date of mailing of the international search report 13 MAR 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer ELLA COLBERT Peggy Harrod Telephone No. (703) 308-7064

Form PCT/ISA/210 (second sheet) (July 1998)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/35169

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,850,442 A (MUFTIC) 15 December 1998, col. 12, lines 59-67, col. 13, lines 1-39, col. 14, lines 3-36, col. 15, lines 19-55, col. 18, lines 11-67, col. 19, lines 1-67, col. 20, lines 1-67, and col. 21, lines 1-18.	1-35

Form PCT/ISA/210 (continuation of second sheet) (July 1998)★